

(19)



Europäisches Patentamt
European Patent Office
Office européen des brevets



(11)

EP 1 072 986 A2

(12)

EUROPEAN PATENT APPLICATION

(43) Date of publication:
31.01.2001 Bulletin 2001/05

(51) Int Cl. 7: G06F 17/30

(21) Application number: 99308130.6

(22) Date of filing: 14.10.1999

(84) Designated Contracting States:

AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU
MC NL PT SE

Designated Extension States:

AL LT LV MK RO SI

(30) Priority: 30.07.1999 US 146621 P

(71) Applicant: Academia Sinica
Nan-Kang, Taipei, 115 (TW)

(72) Inventors:

- Hsu, Chun-Nan, Inst. of Information Science
Nankang, Taipei (TW)
- Chang, Chian-Chi
Wanhua, Taipei (TW)

(74) Representative: Evans, Paul Jonathan et al
Maguire Boss,

5 Crown Street

St. Ives, Cambridge PE27 5EB (GB)

(54) System and method for extracting data from semi-structured text

(57) An information extractor for extracting subsets of text sequences having certain attributes from a semi-structured text sequence. The information extractor resembles a finite state transducer that includes a number of states and transition paths between the states. Each state is defined to be associated with the extraction of subsets of tokens having a certain attribute. Each transition path represents a possible transition from one subset of tokens having a certain attribute to another subset of tokens having another attribute. Contextual rules are used to determine when a transition occurs and which transition path to take. By using contextual rules that consider the token patterns before and after a token, the information extractor can process docu-

ments with irregular layout structures and attributes having a greater variety of permutations.

The state transition diagram, used in one embodiment of the information extractor, represents the different sequence of states that the extractor may go through when it extracts attributes *U,N,A* and/or *M* from an entry in the input text sequence. The extractor starts in state *b/e* 202, then reads a token and determines whether the token belongs to an attribute. For example, the first attribute within an entry must be either *U* or *N*. Hence, there is one transition path 218 from state *b/e* 202 to state *U* 204, and another transition path 220 from state *b/e* 202 to state *N* 208. If the token does not belong to either attribute *U* or *N*, then the extractor remains at state *b/e* 202, shown by the path 224.

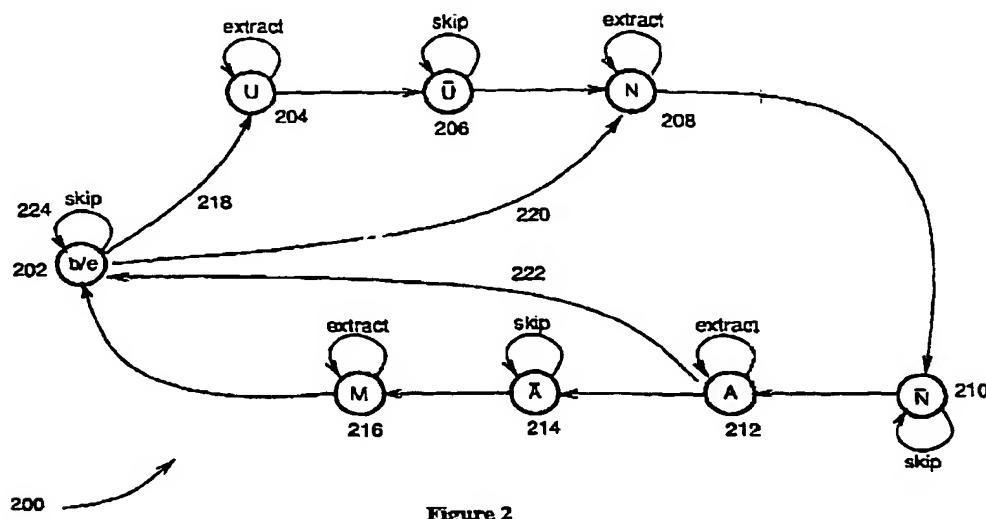


Figure 2

Description**BACKGROUND INFORMATION**

[0001] The present invention relates to extraction of information from text documents.

[0002] Many organizations have large and increasing numbers of electronic files that contain information of great value. The World Wide Web ("Web") itself is a huge depository of such information. Tools that enable computers to search and analyze this vast amount of information are being developed. However, unlike tabular information typically stored in databases today, many documents have only limited internal structure, and thus are called "semi-structured text". For example, a document relating to faculty member information may include a list of entries having words or strings that have certain attributes, such as the faculty name, personal Web page address, contact address, and academic title, etc. The information may be formatted into a tabular list for easy viewing.

[0003] An example of semi-structured text is a Web page written in Hyper Text Markup Language (HTML). HTML uses tags that specify the formats and characteristics of graphics and characters in the Web page. For example, text lying between a pair of tags "" and "" specifies that the text is to be displayed in bold. These HTML tags within the document are directed mainly toward formatting the output style, but the tags themselves may not tell the computer what content is between the tags. For example, given an input string " John ", the tags "" and "" does not indicate to the computer whether "John" corresponds to a name, a contact address, or an academic title.

[0004] A program written to extract information from documents or text sequences is called an information extractor or a "wrapper". An information extractor may be adapted to extract a particular set of words or strings having certain attributes from documents or text sequences that have a particular structure. For example, an information extractor may be adapted to process a Web page such as the one shown in Fig. 1(a) with four entries. Referring to Fig. 1 (b), the HTML text sequence for that Web page has four entries, each entry including a combination of text strings having attributes "URL (*U*)", "Name (*N*)", "Academic Title (*A*)", and "Administrative Title (*M*)". Referring to Fig. 1 (c), the first data tuple to be extracted from the first entry include four text strings having attributes *U*, *N*, *A*, and *M*. The second and third data tuple each contain three text strings having attributes *U*, *N*, and *A*. The fourth data tuple contains text strings having attributes *N* and *A*. For simplicity, a "text string having attribute *U*" will simply be referred to as "attribute *U*". Thus, the term "attribute" will refer to a text sequence having a particular attribute.

[0005] To extract attributes from the text sequence of a Web page, a preprocessor first segments the text sequence into tokens. A token may contain words, num-

bers, HTML tags, etc. For example, in Fig. 1(b), "<H1>" may be a token, while the word "Faculty" may represent another token. An attribute may comprise several tokens. For example, the Name attribute "Mani Chandy"

5 may contains two tokens: "Mani" and "Chandy". The information extractor reads the tokens, and applies a set of predefined extraction rules to extract attributes from these tokens. For example, one possible rule may be that whenever a pair of HTML tags "" and "" are encountered, the text between them is extracted as the Name attribute.

[0006] Extraction rules are typically generated by a separate program called a learning program. A learning program reads a training sample (a short text sequence 15 from the targeted Web page) with attributes labeled by a user. For example, a user may use a graphics user interface (GUI) to highlight or label the parts within the sample text sequence that correspond to the "Name" attribute. The learning program then finds the pattern for 20 locations that the highlighted or labeled part appear in the sample text sequence. That pattern then becomes the rule for extracting the "Name" attribute.

[0007] After a set of extraction rules are generated, a user may test these rules on a second sample text sequence 25 to see if the information extractor can correctly extract all the attributes. If not, then the learning program can be invoked again, and the user labels the attributes that were identified incorrectly with the correct attribute names. The learning program then modifies the 30 extraction rules so that they can correctly identify the attributes in the first and second sample text sequences. The learning program is also used to update the extraction rules when a targeted Web page changes its format or data structure. Because it is not uncommon for Web 35 pages to be modified often, a learning program that requires a minimal set of training samples is desired.

[0008] When extracting attributes from a document, different types of information extractors may use different extraction rules. For example, one information 40 extractor called the "LR-wrapper" uses rules for finding tokens with left-right pair structure. Another information extractor often referred to as the "stalker-wrapper" may have rules for skipping certain types of tokens and searching for certain kinds of "landmark" tokens.

45

SUMMARY

[0009] The present invention is directed to an information extractor that searches for attributes within a 50 semi-structured input text sequence. In general, the information extractor resembles a finite state transducer that includes a number of states and transition paths between the states. Each state is defined to be associated with the extraction of a certain attribute. The allowable 55 transition paths between the states are associated with the possible permutations of the attribute sequences. For each transition path, there is a corresponding contextual rule. When a pattern in the input text sequence

satisfies a particular contextual rule, a transition between states occurs, thus the information extractor enters into the next state that is associated with the extraction of another attribute. By use of contextual rules that also considers the tokens before and after the current input token, the information extractor of the present invention can process documents with irregular layout structures and attributes having a greater variety of permutations.

[0010] The invention features an information extractor that can automatically extract attributes from a greater variety of document styles, and update the extraction rules in an efficient manner. Unlike other information extractors that use a rigid set of rules, and thus can only process documents with attributes arranged with a certain structure the present invention can adapt to variations in document format styles.

[0011] One advantage of the present invention is that the information extractor may be able to identify the attributes correctly even when there are missing attributes, or if the order of attributes varies.

[0012] Another advantage of the present invention is that the information extractor is suitable for processing documents having an irregular data structure or having a tabular layout style.

[0013] An additional advantage of the present invention is that the information extractor use a learning program that requires a minimal set of training samples to generate the extraction rules.

DESCRIPTION OF DRAWINGS

[0014] Fig. 1 (a) shows the output of a sample Web page relating to a faculty and research staff Web site.

[0015] Fig. 1 (b) shows the HTML text sequence (source code) for the Web page of Fig. 1(a).

[0016] Fig. 1(c) shows the attributes extracted from the HTML text sequence of Fig. 1 (b).

[0017] Fig. 2 is an example of a state transition diagram used in one embodiment of the information extractor of the present invention.

[0018] Fig. 3 shows a list of possible transition paths.

[0019] Fig. 4 shows exemplary token types that may be used by an information extractor of the present invention.

[0020] Fig. 5 shows a sample text sequence segmented into tokens using the token types listed in Fig. 4.

[0021] Fig. 6 shows the HTML text sequence (source code) for a sample Web page.

[0022] Fig. 7 shows a state transition diagram used in an alternative embodiment of the information extractor of the present invention.

[0023] Fig. 8 is an example of a state transition diagram used in conjunction with the alternative embodiment of the present invention.

[0024] Fig. 9 (a) shows the HTML text sequence of Fig. 1(b) with arrows indicating the borders of the body containing the attributes.

[0025] Fig. 9 (b) shows the HTML text sequence of Fig. 1(b) with arrows indicating the borders of the attributes.

[0026] Fig. 10 is an example of a tagged-list document.

[0027] Fig. 11 is an example of a state transition diagram used in an embodiment of the present invention for extracting attributes from a tagged-list document.

[0028] Fig. 12 (a) is an example of an attribute list extracted from the tagged-list document of Fig. 10.

[0029] Fig. 12 (b) is an example of a list of *Tag* and *Value* attributes extracted by an information extractor using the state transition diagram of Fig. 11.

[0030] Fig. 13 shows a sample text sequence with attributes enclosed in boxes.

[0031] Fig. 14 is an example of a flow diagram for generating transition rules used by the information extractor of the present invention.

[0032] Fig. 15 (a) is a sample text sequence having tokens *A1*, *A2*, ... *A20*.

[0033] Fig. 15 (b) shows the scope of state *X* and state *N* within the sample text sequence of Fig. 15(a).

[0034] Fig. 15 (c) shows the definition of the left and right context.

[0035] Fig. 16 shows the token class definitions used in an embodiment of the learning program according to the present invention.

[0036] Fig. 17 (a) shows a first set of contextual rules.

[0037] Fig. 17 (b) shows a second set of contextual rules.

[0038] Fig. 17 (c) shows a third set of contextual rules.

[0039] Fig. 18 shows a flow diagram for generating a contextual rule in accordance with an embodiment of the present invention.

[0040] Fig. 19 shows a set of contextual rules for extracting the attributes *Name*, *Academic_title*, and *Admin_title* from a text sequence.

[0041] Fig. 20 shows the sample text sequence of Fig. 13 with the first token of the first attribute of each entry labeled *B1*, *B2*, *B3*, and *B4*.

DETAILED DESCRIPTION

[0042] Referring to Fig. 2, a state transition diagram represents the state transition rules used in an embodiment of the information extractor. In this example, the information extraction is a single pass information extractor suitable for extracting attributes *URL* (*U*), name (*N*), academic title (*A*), and administrative title (*M*) from a text sequence. The input text sequence may come from a Web page such as the one shown in Fig. 1(a), or it may be a document generated by a word processor having similar attributes. The input text sequence is first divided into tokens before being sent to the information extractor. Thus, the information extractor receives the input text sequence along with a sequence of offset numbers indicating the location of each token. The information extractor can extract attributes within

an entry having the sequences of (U, N, A, M) , (U, N, A) , or (N, A) . If an input text sequence has different attributes, or has different permutations of attributes, then the single pass information extractor will also have a different set of state transition rules.

[0043] The term "attribute" is used to refer to a set of consecutive tokens that is a subset of the token sequence having the same characteristics. For example, an attribute U may represent a set of consecutive tokens that form an URL address. A token that is part of the URL address belongs to the attribute U . Similarly, the attribute N may represent a set of consecutive tokens that form a person's name. The term "state" is used as a shorthand or label for describing the actions of an information extractor. The term "extractor" as used herein is intended to mean information extractor. When the extractor is in state U , it means that the extractor is performing functions associated with extracting attribute U . Extracting an attribute may include the actions of reading a sequence of tokens and concatenating them to form a single string. When the extractor enters state U , it means the extractor performing functions associated with reading dummy tokens located between attribute U and the next attribute. A dummy token is a token that does not belong to an attribute.

[0044] When the extractor is in state b/e (abbreviation for begin/end), it means the extractor is reading tokens that are between the last attribute of an entry and the first attribute of the next entry. The state b/e can also mean that the extractor is performing functions associated with reading the dummy tokens before the first attribute in the input text sequence. The state b/e may also mean that the extractor is performing functions associated with reading the dummy tokens after the last attribute in the input text sequence. In certain applications, a **GB** (abbreviation for Global Begin) state is defined to be associated with the actions of reading the tokens before the first attribute. Likewise, a **GE** (abbreviation for Global End) is defined to be associated with the actions of reading the tokens after last attribute in the text sequence.

[0045] The term "contextual rule" refers to comparing the context of a token with a set of predetermined token patterns to see if there is a match. The context of a token includes the token under consideration, and possibly the tokens before and/or after it. For example, suppose the input text sequence is:

`<DT> ` and the token "http" identifies a transition from state b/e to state U , a contextual rule may be:

Transfer from state b/e to state U if

Left context = "HREF=s"" and

Right context = "http://".

[0046] The reference point for the left and right context is the location immediately preceding the token under consideration ("http"). Thus, the "left context" refers to the token(s) before "http", and the "right context" refers to "http" and a few additional tokens -- ":"; "/", and "/".

". The specific number of tokens in the left context and the right context varies with the contextual rules. For example, a contextual rule may require the left context to include 3 tokens and the right context to include only one token. Another contextual rule may require the right context to include two tokens, while not requiring a comparison of the left context.

[0047] Within an entry, we define a dummy attribute U' to include all tokens that are between attribute U and the next attribute. Likewise, we define a dummy attribute N' to include all the dummy tokens between attribute N and the attribute A . Dummy attribute A' includes dummy tokens between attribute A and M .

[0048] The state transition diagram in Fig. 2 represents the different sequence of states that the extractor may go through when it extracts attributes U , N , A , and/or M from an entry in the input text sequence. The extractor starts in state b/e 202, then reads a token and determines whether the token belongs to an attribute.

[0049] Referring to Fig. 3, the possible state transitions for the extractor defined by the state transition diagram of Fig. 2 is shown. For each possible state transition listed in Fig. 3, there is a corresponding transition path in Fig. 2. The transition from state U 204 to state U' 206 means that the end of attribute U is detected, and that the current token (and possibly the tokens afterwards) is a dummy token that do not belong to any attribute. The transition from state U' 206 to state N 208 means that the beginning of state N 208 is detected, and that the current token (and possibly the tokens afterwards) belongs to attribute N . When the information extractor extracts attributes U , N , and A from the second entry in Figure 1(b), it goes through the states $b/e - U - U' - N - N' - A - b/e$.

[0050] After the last attribute (A) within an entry is extracted, the extractor transfers to state b/e 202, meaning that the current token is a dummy token located before the first attribute of the next entry.

[0051] There are two transition paths leaving state b/e 202 in Fig. 2. The extractor determines which path to take by use of contextual rules. Each path has a corresponding contextual rule. If the context of a token matches the contextual rule of a transition path, then the state is transferred according to that contextual rule. For example, the contextual rule associated with the transition of state b/e 202 to U 204 may be

Rule 1: Transfer from state b/e to state U if

Left context = Calph(HREF) Punc(=) Punc

(") and

Right context = Oalph(http) Punc (:) Punc(/)

Punc(/).

(Note: The token class Calph() represents strings having all capital letters. The token class Punc () represents

punctuation marks. The token class **Oalph()** represents strings having all lower case letters. These are described in more detailed in Fig. 4.) Rule 1 requires that the left context to be the string "HREF="" , and the right context to be the string "http://". The contextual rule associated with the transition of state **b/e** 202 to **N** 208 may be

Rule 2: Transfer from state **b/e** to state **N** if
 Left context = **Html()** and
 Right context = **C1alph(_)**.

(Note: The token class **Html()** represents all HTML tags, and the token class **C1alph()** represents strings that begin with a capital letter and has at least one lower case letter. The symbol "_" denotes the wildcard character, thus any string that begins with a capital letter and has at least one lower case letter will match the token pattern **C1alph(_)**.)

[0051] If the context of a token satisfies Rule 1, then the transition path from state **b/e** 202 to state **U** 204 is taken. On the other hand, if the context of a token satisfies Rule 2, then the transition path from state **b/e** 202 to state **N** 208 is taken. Otherwise, the extractor remains in state **b/e** 202 and continues reading tokens until one of the two rules is satisfied.

[0052] In some applications, it may be feasible to add a **GB** state before the first **b/e** state, and a **GE** state after the last **b/e** state. This is because the left/right context of the first token of the first attribute in the first entry may be different from the left/right context of the first token of the first attribute in the second entry. Likewise, the left/right context of the last token of the last attribute in the last entry may be different from the left/right context of the last token of the last attribute in the second-to-last entry.

[0053] Referring to Fig. 4, the token classes used to tokenize a text sequence are described. The first column lists the name of the token class, the second column lists the pattern of the token, the third column describes the token type, and the last column lists the length of the token. Other token classes may be defined depending on the particular application. The learning program described later may use different token classes to generate the left/right context patterns.

[0054] Referring to Fig. 5, the text sequence of the second line of the HTML source code shown in Fig. 1 (b) is segmented into tokens. The first token is an HTML tag "<DT>", the second token is a space character " ", and the third token is an HTML tag "<A>", etc. The eleventh token is the punctuation "/", the twelfth token is the string "www", and the thirteenth token is the punctuation ".", etc.

[0055] The left context of token 1 ("<DT>") is a new line character (which is the end of the previous line in Fig. 1(b), and is not shown in Fig. 5). Thus, token 1 does not satisfy Rule 1 or Rule 2. Tokens 2, 3, 4, 5, 6, and 7 also do not satisfy Rules 1 and 2. Token 8 satisfies Rule 1 because the left context of token 8 is "HREF="" , and the right context of token 8 is "http://". Therefore, at to-

ken 8, the extractor transfers from state **b/e** 202 to state **U** 208. Note that under our definition, the separation point of left/right context for token 8 is between the quotation mark ("") and the character 'h'.

5 [0056] The contextual rules require that the extractor "look-ahead" and read additional tokens in order to make a comparison of the right context. For example, Rule 1 requires that the extractor read an additional three tokens to make the comparison of the right context because "http://" includes four tokens. The specific number of additional tokens that need to be read depends on the particular set of contextual rules. Initially, the extractor may read a number of tokens required by the contextual rules.

10 [0057] Another example of a contextual rule that identifies a transition from any other state to state **A** may be:

Rule 3: Transfer to state **A** if

Left context = **Html(<DD>)** **Html(<I>)** and
 Right context = **C1alph(_)**

20 Rule 3 requires that the left context is an HTML tag "<DD>" followed by another HTML tag "<I>". In addition, the right context must be a string that begins with a capital letter and has at least one lower case letter.

[0058] Referring to Fig. 6, a Web page having content and layout format slightly different from the Web page of Fig. 1 (a) has a slightly different HTML text sequence (or source code) from that in Fig. 1(b). Accordingly, an extractor will use different contextual rules to identify the transitions between states associated with extracting different attributes from the Web page. The contextual rule for transition from any other state to state **A** in this example may be:

Rule 4: Transfer to state **A** if

Left context = (**Html()** **Punc(,)** **Spc(_)** **Html**

35 (**<I>**) or
Punc(_) **NL(_)** **Spc(_)** **Html(<I>)** or
Punc(,) **Spc(_)** **Html(<I>)** and
 Right context = **C1alph(_)**

Rule 4 requires that the left context is either (i) an HTML tag "", followed by a comma, any number of spaces, and an HTML tag "<I>"; or (ii) any punctuation mark, followed by a new line character, any number of spaces, and an HTML tag "<I>"; or iii) a comma, followed by any number of spaces, and an HTML tag "<I>". Rule 4 requires that the right context is a string that begins with a capital letter, and has at least one lower case letter.

[0059] Referring to Fig. 7, a state transition diagram 700 represents the state transition rules used by another embodiment of the information extractor. This is a universal single pass information extractor that has states **b/e**, **U**, **N**, **A**, **M**, and **D**. It is suitable for extracting attributes **U**, **N**, **A**, and **M** from an input text sequence. The input text sequence may come from a Web page such as the one shown in Fig. 1(a) or Fig. 6, or it may be a document generated by a word processor having similar attributes.

50 [0060] This extractor can process documents having attributes **U**, **N**, **A**, and **M** that appear in any permutation

sequence. For example, the extractor can process an entry that has attributes (M, A, U, N) . Such a sequence cannot be processed by the single pass extractor described in accordance with Fig. 2. The diagram in Fig. 7 is used only as an example. If an input text sequence has different attributes, then the universal single pass information extractor will also have a different set of state transition rules.

[0061] The extractor can process an entry that has attributes (U, N, M) by traversing the states **b/e - U - D - N - D - M - b/e**. When the extractor is in state **D**, the extractor is reading dummy tokens that are located between attributes. The extractor starts with state **b/e** 702. The extractor reads sequential ones of tokens and remains in state **b/e** 702 via path 726 until a token belonging to attribute **U** is identified. Then the extractor transfers to state **U** 704 via path 712. The extractor reads tokens and remains in state **U** 704 via path 728 until a token that does not belong to attribute **U** is identified. The extractor stores (or outputs) the extracted attribute **U**, then transfers to state **D** 712 via path 714. The extractor reads tokens and remains in state **D** 712 via path 730 until a token that belongs to attribute **N** is identified. The extractor transfers to state **N** 706 via path 716, extracts the attribute **N**, then transfers to state **D** 712 again via path 718. Next, the extractor goes to state **M** 710 via path 722, extracts attribute **M**, then goes to state **b/e** 702 via path 724, ending the cycle of extraction for this entry.

[0062] Referring to Fig. 8, a state transition diagram 800 represents the state transition rules used by another embodiment of the information extractor. This is a multi-pass information extractor suitable for extracting attributes **U, N, A**, and **M** in an input text sequence. If an input text sequence has different attributes, then the multi-pass information extractor will also have a different set of state transition rules.

[0063] An information extractor utilizing the transition rules of Fig. 8 has the equivalent of six information extractors 802, 804, 806, 808, 810, and 812, each performing partial extraction on the input text sequence. The extractor 802 scans through the input text sequence and finds the part of the input document, referred to as the "body"; that starts with the first attribute and ends with the last attribute.

[0064] Referring to Fig. 9 (a), the first arrow pointing to "h" at line 2 shows the beginning of the first attribute, and the arrow pointing to "e" at line 10 shows the end of the last attribute. The extractor 802 finds the offset values indicated by the first and second arrows in Fig. 9 (a), and sends these two offset values along with the input text sequence to the extractor 804.

[0065] The state **GB814** in Fig. 8 represents the state in which the extractor is performing functions associated with reading dummy tokens before the first attribute. The state **Body816** represents the state in which the extractor is performing functions associated with reading tokens within the body part, which includes the first and

last attribute and all the text sequence in between. The extractor enters state **GE818** after the last token of the last attribute has been read. One contextual rule, designated as **(GB, Body)**, is associated with the transition path 824 from state **GB814** to state **Body816**. Another contextual rule, designated as **(Body, GE)**, is associated with the transition path 826 from state **Body816** to state **GE818**.

[0066] The extractor 802 in Fig. 8 starts in state **GB814**. It reads a token and compares the context of the token with the contextual rule **(GB, Body)**. Because the first attribute of the input text sequence is either attribute **U** or attribute **N**, the contextual rule **(GB, Body)** actually comprises of the rules **(GB, U)** and **(GB, N)**. The rule **(GB, U)** identifies the transition between state **GB814** to state **U** (not shown here). The rule **(GB, N)** identifies the transition from state **GB814** to state **N** (not shown here). If either rule **(GB, U)** or rule **(GB, N)** is satisfied, then the extractor 802 enters state **Body816**; otherwise, it remains in state **GB814**. After the extractor enters the state **Body816**, it stores the first token and the offset value of the start of the first token. It then reads the next token and compares the context of that token with the contextual rule **(Body, GE)**. If the rule **(Body, GE)** is not matched, the extractor 802 stores the token, remains in state **Body816**, and continues to read the next token. If rule **(Body, GE)** is satisfied, then the extractor 802 enters state **GE818**, stores the offset value of the end of the last attribute, and ends the process for extractor 802. The extractor 804 then carries on the extraction process.

[0067] Referring to Fig. 9 (b), the arrows indicate the start and end of attributes, representative of the offset values generated by the extractor 804. The attributes are extracted when the extractor 804 is in the state **Tuple820**. The state **Tuple822** refers to the state when the extractor 804 is reading dummy tokens located between attributes. The extractor 804 receives the input text sequence and the two offset values generated by the extractor 802, and repeatedly extracts the attributes without identifying whether it is attribute **U, N, A**, or **M**. The output of the extractor 804 is the offset values of the start and end of all attributes. After finding the offset values, the four extractors 806, 808, 810, and 812 are invoked to extract attributes **U, N, A**, and **M**, respectively.

[0068] The single pass and multi-pass information extractors of the present invention can process documents having missing attributes. For example, one faculty member may not have an administrative title, while another may not have an URL address. The extractors of the present invention can also process documents having multiple attributes, e.g., a faculty may have two or more administrative titles. The extractors may also process documents having variant attribute ordering. For example, a Web page may have several entries with the academic title appearing before the administrative title, but also has one entry which the academic title appears after the administrative title.

[0069] Referring to Fig. 10, a tagged-list text sequence 1000 has five entries. Each entry includes attributes "Name", "E-Mail", "Last Update", "Alt. Name", "Organization", and/or "Service Provider". The text sequence 1000 is different from the text sequence in Fig. 1(b) because text sequence 1000 contains the attribute names (herein referred to as "tags") for each attribute (herein referred to as "tag values"). An extractor can extract the attribute names along with the attributes ("tag-and-value pairs"), and then apply a post-processing step to convert the tag-and-value pairs into data tuples of attribute values.

[0070] Referring to Fig. 11, a state transition diagram 1100 represents the state transition rules used by an embodiment of the information extractor for extracting a tagged-list text sequence. The extractor is suitable for extracting attributes *Tag* and *Value* (*Val*). When the extractor is in state *Tag* 1102, it is extracting tokens that belong to attribute *Tag*, such as "Name" or "E-Mail". When the extractor is in state *Val* 1106, it is extracting tokens that belong to attribute *Value*, such as "Lithium' J Smith", or "aulmer@u.Washington.edu", etc. When the extractor is in state *Tag*' 1104 and state *Val*' 1108, it is extracting dummy tokens.

[0071] Referring to Fig. 12 (a), the attributes sought to be extracted from the first entry in the text sequence of Fig. 10 are "Name", "E-Mail", "Last-Update", and "Organization". Fig. 12 (b) shows the *Tag* and *Value* attributes extracted by an information extractor using the state transition diagram of Fig. 11. Note that the extractor does not differentiate between *Tag* or *Value* attributes that have different contents. A post processor can then transform the *Tag* and *Value* output in Fig. 12 (b) into the attribute list format in Fig. 12(a).

Generating the Transition Paths

[0072] In accordance to the present invention, a learning program is provided to generate the transition paths used by an information extractor. In one embodiment, a sample text sequence representative of a target document is first given to the learning program. A user highlights the attributes within the sample text sequence, and then identifies which highlighted part is associated with which attribute. In the description hereafter, the HTML sequence in Fig. 1 (b) will be used as the sample text sequence.

[0073] Referring to Fig 13, a user first uses a GUI to highlight the attributes on a display screen. Here, the attributes are enclosed in boxes. Then the learning program interactively prompts the user to enter the attribute name for each highlighted part. From these highlighted parts and the attribute names provided by the user, the learning can identify the different attribute combinations within an entry. For example, the learning program may identify the sample text sequence to have four entries, each entry having the data tuple (*U*, *N*, *A*, *M*), (*U*, *N*, *A*), (*U*, *N*, *A*), and (*N*, *A*), respectively. Other methods for

labeling the attributes can be used to accommodate the particular GUI or other kinds of input device.

[0074] Referring to Fig. 14, a flow diagram 1400 represents the process of generating a set of transition paths. The learning program starts at block 1402. At block 1404, the learning program checks the first attribute of each data tuple. In this example, the first attribute is either *U* or *N*. For each such attribute *i* that is the first attribute of a data tuple, a transition path is created from state *b/e* to state *i*. Thus, transition paths (*b/e* → *U*) and (*b/e* → *N*) are created.

[0075] At block 1406, the learning program finds all possible attributes pairs that come in sequence. The possible attribute pairs are (*U*, *N*), (*N*, *A*), and (*A*, *M*). At block 1406, for each attribute pair (*j*, *k*) that comes in sequence, a transition path is created from state *j* to state *j'*, and from state *j'* to state *k*. State *j'* is a dummy state, and represents the state in which the extractor is reading a dummy token after the attribute *j*. The transition paths (*U* → *U'*), (*U'* → *N*), (*N* → *N'*), (*N'* → *A*), (*A* → *A'*), (*A'* → *M*) are created in block 1406.

[0076] At block 1408, the learning program checks to see which is the last attribute for each data tuple. Here, the last attribute can be either *A* or *M*. At block 1408, for each such last attribute *m*, a transition path is created from state *m* to state *b/e*. The transition paths (*A* → *b/e*) and (*M* → *b/e*) are created at block 1408. The transition path generation process is ended at block 1410. A total of 10 transition paths are created in this example. These transition paths are consistent with the ones shown in Fig. 3.

Generating the Contextual Rules

[0077] In accordance with the present invention, the learning program can further generate the contextual rules used to determine when to enter or leave a state. The contextual rules are also used to determine which transition path to take when there are multiple transition paths branching out of a state. A sample text sequence with attributes correctly labeled by a user is given to the learning program. The learning program then generates the contextual rules by using a "set-covering" algorithm to cover all "positive token examples" and exclude all "negative token examples". Positive token examples are tokens such that their context should match the contextual rules. Negative token examples are tokens such that their context should not match the contextual rules.

[0078] Referring to Fig. 15 (a), a sample text sequence having tokens *A*1, *A*2, ... *A*20 is used to illustrate the method of generating a contextual rule. The text sequence has a *Name* attribute "Yaser Abu-Mostafa" that includes tokens *A*16 to *A*18. The tokens *A*1 to *A*15, and *A*19, *A*20 are dummy tokens.

[0079] Referring to Fig. 15 (b), the extractor is defined to be in state *X* when reading dummy tokens before attribute *Name*, and be in state *N* when reading tokens that belong to attribute *Name*. The contextual rule for

identifying the transition from state X to state N can be written as:

Rule 5: Transfer from state X to state N if

Left context = LP (left context pattern) and
Right context = RP (right context pattern).

[0080] The goal is to find the left context pattern (LP) and the right context pattern (RP). LP/RP should only match the left/right context of token $A16$, and should not match the left/right context of tokens $A1$ to $A15$. Token $A16$ is an example of a "positive sample"; and tokens $A1$ to $A15$ are examples of "negative samples". A positive sample is a token that the contextual rule should correctly match, and a negative sample is a token that the contextual rule should not match. In general, there may be more than one positive sample. The contextual rule should match each one of the positive samples, and not match any of the negative samples. In this example, the goal is to find LP/RP such that when the extractor applies Rule 5 to tokens $A1$ to $A16$, the extractor can correctly identify the transition from state X to state N at token $A16$.

[0081] Referring to Fig. 15 (c), the first token of the right context of token $A16$ is $A16$; the second token of the right context of $A16$ is $A17$, and so on. The first token of the left context of token $A16$ is $A15$; the second token of the left context of $A16$ is $A14$, and so on.

[0082] Initially, the learning program takes the first token of the left context of $A16$ to generate a left context pattern. The left context pattern is a token class that includes the first token of the left context of $A16$. Because $A15$ is an HTML tag "", the left context pattern for the first token of the left context can be $Html(_)$ or $Ptag(_)$. The token classes used by the learning program are listed in Fig. 16.

[0083] Referring to Fig. 17 (a), a tentative contextual rule may be:

Rule 6: Transfer from state X to state N if

Left context = $Html(_)$.

Any token having an HTML tag as first token of the left context will satisfy Rule 6. Applying Rule 6 to tokens $A1$ to $A15$, the token pattern $Html(_)$ matches the left context of negative samples $A2$, $A10$, $A13$, and $A15$. The positive matching count (p) is defined as the number of positive samples that are matched, and the negative matching count (n) is defined as the number of negative samples that are matched. Here, $p = 1$ and $n = 4$.

[0084] Because "" is also a member of the token class $Ptag(_)$, another tentative contextual rule may be:

Rule 7: Transfer from state X to state N if

Left context = $Ptag(_)$.

The token pattern $Ptag(_)$ matches the left context of negative samples $A2$, $A10$, $A13$, and $A15$. Thus $p = 1$ and $n = 4$.

[0085] The learning program then takes the first token of the right context of $A16$ to generate a right context pattern. In this example, $A16$ is a word "Yaser", thus the right context pattern for the first token of the right context may be $Word(_)$, $CNalph(_)$, or $Clalph(_)$. These three

token classes all match the token "Yaser".

[0086] Rule 8 uses $Word(_)$ as the right context pattern. Comparing $Word(_)$ with the right context of tokens $A1$ to $A15$, $Word(_)$ matches the right context of negative samples $A2$, $A4$, $A6$, and $A8$. Thus $p = 1$ and $n = 4$. For Rule 9, $CNalph(_)$ also matches the right context of four negative samples, thus $p = 1$ and $n = 4$. For Rule 10, $Clalph(_)$ matches the right context of negative samples $A2$, $A6$, and $A8$, thus $p = 1$ and $n = 3$.

[0087] Comparing Rules 6 to 10, Rule 10 results in the fewest incorrect matches (matching a negative sample). Typically, the learning program selects the tentative contextual rule that results in the greatest value for $(p - n) / (p + n)$. If applying two rules to the relevant tokens result in the same values for $(p - n) / (p + n)$, then the rule having a token class that is broader will be selected as the basis for finding the complete contextual rule. For example, the learning program will choose Rule 8 over Rule 9 because $Word(_)$ has a broader scope than $CNalph(_)$. The context pattern used in Rule 10 is incomplete because it still matches some negative samples.

[0088] Referring to Fig. 17 (b), the learning program finds the next set of tentative contextual rules (Rules 11 - 15) by using $Clalph(_)$ as basis, and adding one token pattern to either the left or right context pattern. For Rules 11, $Html(_)$ is added to the left context pattern of Rule 10. Thus, Rule 11 becomes:

Rule 11: Transfer from state X to state N if

Left context = $Html(_)$ and

Right context = $Clalph(_)$.

At this point, it is only necessary to compare $Html(_)$ with the first token of the left context of tokens of $A2$, $A6$, and $A8$. Here, $Html(_)$ matches the first token of the left context of $A2$, thus $p = 1$ and $n = 1$. For Rule 12, $Ptag(_)$ also matches the first token of the left context of $A2$, thus $p = 1$ and $n = 1$.

[0089] For Rules 13 to 15, the learning program expands the right context pattern by adding a token pattern that matches the second token of the right context of $A16$. The second token of the right context of $A16$ is the space character " ", which belongs to the token classes $Nonword(_)$, $Ctrl(_)$, and $Spc(_)$. For Rule 13, $Nonword(_)$ is added to the right context pattern of Rule 10. Thus, Rule 13 becomes:

Rule 13: Transfer from state X to state N if

Right context = $Clalph(_)$ $Nonword(_)$.

The learning program compares $Nonword(_)$ with the second token of the right context of the negative samples matched by Rule 10. Here, $Nonword(_)$ matches the second token of the right context of $A2$ and $A6$, thus $p = 1$ and $n = 2$. For Rule 14, $Ctrl(_)$ matches the second token of the right context of $A2$ and $A6$, thus $p = 1$ and $n = 2$. For Rule 15, $Spc(_)$ also matches the second token of the right context of $A2$ and $A6$, thus $p = 1$ and $n = 2$. Comparing Rules 11 to 15, Rule 11 and 12 results in the fewest incorrect matches. Because $Html(_)$ is a broader class than $Ptag(_)$, the learning program se-

lects Rule 11 as basis for finding the complete context pattern for the contextual rule.

[0090] Referring to Fig. 17 (c), the learning program finds the next set of tentative contextual rules by using $LP = \text{Html}(_)$ and $RP = \text{C1alph}(_)$ as basis, and adding one token pattern to either the left or right context patterns. The second token of the left context of $A16$ is $<\text{A}\text{ HREF}=\text{"http://electra.caltech.edu/EE/Faculty/Abu-Mostafa.html"}>$. This token belongs to the token classes $\text{Html}(_)$ and $\text{Ptag}(_)$. Rule 16 is generated by adding $\text{Html}(_)$ to the left context pattern of Rule 11. Rule 17 is generated by adding $\text{Ptag}(_)$ to the left context pattern of Rule 11. Rule 16 is:

Rule 16: Transfer from state X to state N if
 Left context = $\text{Html}(_)$ $\text{Html}(_)$ and
 Right context = $\text{C1alph}(_)$.

The learning program compares $\text{Html}(_)$ to the second token of the left context of the negative samples matched by Rule 11. Here, $\text{Html}(_)$ does not match any second token of the left context of the negative samples. Thus, $p = 1$ and $n = 0$. For Rule 17, $\text{Ptag}(_)$ also does not match any second token of the left context of the negative samples, thus $p = 1$ and $n = 0$. Applying Rules 18 - 20 all result in $p = 1$ and $n = 1$. Thus, Rule 16 and 17 are better than Rules 18 - 20. Rule 16 is selected as the final contextual rule because $\text{Html}(_)$ is a broader class than $\text{Ptag}(_)$. Therefore, the contextual rule for identifying the transition from state X to state A is Rule 16.

[0091] Fig. 18 illustrates a flow diagram for generating a contextual rule in accordance with an embodiment of the present invention. The learning program starts at block 1802, and receives input from a user in block 1804 to identify positive and negative samples that identify transitions from one state to another. An example of a positive sample for the transition from state X to state N is the token $A16$ in Fig. 15 (c), and examples of negative samples are tokens $A1$ to $A15$. In block 1806, the learning program generates a series of left context patterns that match the first token of the left context of the positive samples. Examples of such left context patterns are $\text{Html}(_)$ and $\text{Ptag}(_)$. In block 1808, the learning program generates a series of right context patterns that match the first token of the right context of the positive samples. Examples of such right context patterns are $\text{Word}(_), \text{CNalph}(_),$ and $\text{C1alph}(_)$.

[0092] In block 1810, the learning program compares the left context patterns generated in block 1806 with the first token of the left context of negative samples, and determines the n value representing the number of negative samples that are matched. The learning program then compares the right context patterns generated in block 1808 with the first token of the right context of negative samples, and determines the n value representing the number of negative samples that are matched. The learning program chooses the context pattern that results in the largest value for $(p - n)/(p + n)$. The number p represents the number of positive sam-

ples.

[0093] In block 1812, the learning program determines whether the context patterns generated so far is complete. If the left/right context pattern selected in block 1810 does not match any negative samples, then that context pattern is complete, and control is transferred to block 1816, and output the contextual rule using the left/right context patterns generated thus far. In block 1812, if the left/right context pattern selected in block 1810 still matches any negative samples, then control is transferred to block 1814. The left (or right) context pattern is expanded by adding a token pattern that matches the next token of the left (or right) context of the positive samples. Typically, block 1814 requires generating a series of context patterns and finding the one that results in the largest $(p - n)/(p + n)$ value, similar to the actions performed in blocks 1806, 1808, and 1810. Blocks 1812, 1814 and 1810 are repeated until a left/right context pattern that doesn't match any negative sample is generated.

[0094] Referring to Fig. 19, Rules 21 - 28 form a sample set of contextual rules that can be used by an information extractor to extract the attributes "Name", "Academic_title", and "Admin_title" from a text sequence similar to the one shown in Fig. 1(b). Rule 21 is used to identify the transition from state GB to state b/e . Rule 22 is used to identify the transition from state b/e to state Name , and so on. The information extractor using Rules 21 - 28 has eight states: GB , b/e , Name , $@\text{Name}$, Academic_title , $@\text{Academic_title}$, Admin_title , and GE . The state $@\text{Name}$ refers to a dummy state in which the extractor is reading dummy tokens between attribute Name and the next attribute. Each one of the Rules 21 - 28 can be generated by the learning program in accordance with the flow diagram of Fig. 18.

[0095] Referring to Fig. 20, the sample text sequence of Fig. 13 is shown with each attribute highlighted. For purpose of illustration, the first token of the first attribute of the four entries are labeled B1, B2, B3, and B4, respectively. Note that tokens B1, B2, and B3 have already been identified as belonging to attribute U , and B4 as belonging to attribute N . The description below illustrates the process of generating the contextual rules for an information extractor having the state transition diagram of Fig. 2. In Fig. 2, there are paths leaving state b/e : transition paths $(b/e \rightarrow U)$ and $(b/e \rightarrow N)$.

[0096] To generate the contextual rule for the transition path $(b/e \rightarrow U)$, the learning program identifies tokens B1, B2, and B3 as positive samples because they belong to attribute U . All other tokens as negative samples. The learning program generates left/right context patterns that match only tokens B1, B2, and B3, but no other tokens. To generate the contextual rule for the transition path $(b/e \rightarrow N)$, the learning program first identifies token B4 as a positive sample because it belongs to attribute N . All other tokens as negative samples. The learning program then generates left/right context pat-

terns that match only token B4, but no other tokens. The contextual rules with other transition paths can be generated in a similar way.

[0097] The presently disclosed embodiments are considered in all respects to be illustrative and not restrictive. The scope of the invention is indicated by the appended claims rather than the foregoing description, and all changes that come within the meaning and range of equivalents thereof are intended to be embraced therein.

Claims

1. A method of extracting attributes from a token sequence, the method comprising the steps of:

identifying the beginning of a first attribute from the token sequence by applying a first set of contextual rules to sequential ones of the token sequence including comparing a left context and a right context of a token of the token sequence with a set of predetermined token patterns to determine whether one of said set of predetermined token patterns is satisfied; upon identifying the beginning of the first attribute, storing sequential tokens which follow the token associated with the beginning of the first attribute until the end of the first attribute is identified.

2. The method of claim 1 wherein identifying the end of the first attribute includes applying a second set of contextual rules to sequential ones of the token sequence after the token associated with the beginning of the first attribute.

3. The method of claim 2, after the step of identifying the end of the first attribute, further comprising the steps of:

identifying the beginning of a second attribute from the token sequence by applying a third set of contextual rules to sequential ones of the token sequence after the token associated with the end of the first attribute, including comparing a left context and a right context of a token of the token sequence with a set of predetermined token patterns to determine whether one of said set of predetermined token patterns is satisfied; upon identifying the beginning of the second attribute, storing sequential tokens which follow the token associated with the beginning of the second attribute until the end of the second attribute is identified; and identifying the end of the second attribute including applying a fourth set of contextual rules

to sequential ones of the token sequence after the token associated with the beginning of the second attribute.

5. 4. The method of claim 3, wherein the first and second attributes belong to a predetermined set of attributes having a predetermined number of permutation sequences, and the set of predetermined token patterns used in said third set of contextual rules include token patterns for identifying a transition from a token not associated with any attribute to a token associated with an attribute that can possibly follow the first attribute according to the predetermined set of permutation sequences.
- 15 5. The method of claim 3, wherein the first and second attributes belong to a predetermined set of attributes having a predetermined number of permutation sequences, and the set of predetermined token patterns used in said third set of contextual rules include token patterns for identifying a transition from a token not associated with any attribute to a token associated with any attribute belonging to the predetermined set of attributes.
- 20 6. A method of extracting a first subset of tokens having a first attribute and a second subset of tokens having a second attribute from a token sequence, the method comprising the steps of:
- 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120 125 130 135 140 145 150 155 160 165 170 175 180 185 190 195 200 205 210 215 220 225 230 235 240 245 250 255 260 265 270 275 280 285 290 295 300 305 310 315 320 325 330 335 340 345 350 355 360 365 370 375 380 385 390 395 400 405 410 415 420 425 430 435 440 445 450 455 460 465 470 475 480 485 490 495 500 505 510 515 520 525 530 535 540 545 550 555 560 565 570 575 580 585 590 595 600 605 610 615 620 625 630 635 640 645 650 655 660 665 670 675 680 685 690 695 700 705 710 715 720 725 730 735 740 745 750 755 760 765 770 775 780 785 790 795 800 805 810 815 820 825 830 835 840 845 850 855 860 865 870 875 880 885 890 895 900 905 910 915 920 925 930 935 940 945 950 955 960 965 970 975 980 985 990 995 1000 1005 1010 1015 1020 1025 1030 1035 1040 1045 1050 1055 1060 1065 1070 1075 1080 1085 1090 1095 1100 1105 1110 1115 1120 1125 1130 1135 1140 1145 1150 1155 1160 1165 1170 1175 1180 1185 1190 1195 1200 1205 1210 1215 1220 1225 1230 1235 1240 1245 1250 1255 1260 1265 1270 1275 1280 1285 1290 1295 1300 1305 1310 1315 1320 1325 1330 1335 1340 1345 1350 1355 1360 1365 1370 1375 1380 1385 1390 1395 1400 1405 1410 1415 1420 1425 1430 1435 1440 1445 1450 1455 1460 1465 1470 1475 1480 1485 1490 1495 1500 1505 1510 1515 1520 1525 1530 1535 1540 1545 1550 1555 1560 1565 1570 1575 1580 1585 1590 1595 1600 1605 1610 1615 1620 1625 1630 1635 1640 1645 1650 1655 1660 1665 1670 1675 1680 1685 1690 1695 1700 1705 1710 1715 1720 1725 1730 1735 1740 1745 1750 1755 1760 1765 1770 1775 1780 1785 1790 1795 1800 1805 1810 1815 1820 1825 1830 1835 1840 1845 1850 1855 1860 1865 1870 1875 1880 1885 1890 1895 1900 1905 1910 1915 1920 1925 1930 1935 1940 1945 1950 1955 1960 1965 1970 1975 1980 1985 1990 1995 2000 2005 2010 2015 2020 2025 2030 2035 2040 2045 2050 2055 2060 2065 2070 2075 2080 2085 2090 2095 2100 2105 2110 2115 2120 2125 2130 2135 2140 2145 2150 2155 2160 2165 2170 2175 2180 2185 2190 2195 2200 2205 2210 2215 2220 2225 2230 2235 2240 2245 2250 2255 2260 2265 2270 2275 2280 2285 2290 2295 2300 2305 2310 2315 2320 2325 2330 2335 2340 2345 2350 2355 2360 2365 2370 2375 2380 2385 2390 2395 2400 2405 2410 2415 2420 2425 2430 2435 2440 2445 2450 2455 2460 2465 2470 2475 2480 2485 2490 2495 2500 2505 2510 2515 2520 2525 2530 2535 2540 2545 2550 2555 2560 2565 2570 2575 2580 2585 2590 2595 2600 2605 2610 2615 2620 2625 2630 2635 2640 2645 2650 2655 2660 2665 2670 2675 2680 2685 2690 2695 2700 2705 2710 2715 2720 2725 2730 2735 2740 2745 2750 2755 2760 2765 2770 2775 2780 2785 2790 2795 2800 2805 2810 2815 2820 2825 2830 2835 2840 2845 2850 2855 2860 2865 2870 2875 2880 2885 2890 2895 2900 2905 2910 2915 2920 2925 2930 2935 2940 2945 2950 2955 2960 2965 2970 2975 2980 2985 2990 2995 3000 3005 3010 3015 3020 3025 3030 3035 3040 3045 3050 3055 3060 3065 3070 3075 3080 3085 3090 3095 3100 3105 3110 3115 3120 3125 3130 3135 3140 3145 3150 3155 3160 3165 3170 3175 3180 3185 3190 3195 3200 3205 3210 3215 3220 3225 3230 3235 3240 3245 3250 3255 3260 3265 3270 3275 3280 3285 3290 3295 3300 3305 3310 3315 3320 3325 3330 3335 3340 3345 3350 3355 3360 3365 3370 3375 3380 3385 3390 3395 3400 3405 3410 3415 3420 3425 3430 3435 3440 3445 3450 3455 3460 3465 3470 3475 3480 3485 3490 3495 3500 3505 3510 3515 3520 3525 3530 3535 3540 3545 3550 3555 3560 3565 3570 3575 3580 3585 3590 3595 3600 3605 3610 3615 3620 3625 3630 3635 3640 3645 3650 3655 3660 3665 3670 3675 3680 3685 3690 3695 3700 3705 3710 3715 3720 3725 3730 3735 3740 3745 3750 3755 3760 3765 3770 3775 3780 3785 3790 3795 3800 3805 3810 3815 3820 3825 3830 3835 3840 3845 3850 3855 3860 3865 3870 3875 3880 3885 3890 3895 3900 3905 3910 3915 3920 3925 3930 3935 3940 3945 3950 3955 3960 3965 3970 3975 3980 3985 3990 3995 4000 4005 4010 4015 4020 4025 4030 4035 4040 4045 4050 4055 4060 4065 4070 4075 4080 4085 4090 4095 4100 4105 4110 4115 4120 4125 4130 4135 4140 4145 4150 4155 4160 4165 4170 4175 4180 4185 4190 4195 4200 4205 4210 4215 4220 4225 4230 4235 4240 4245 4250 4255 4260 4265 4270 4275 4280 4285 4290 4295 4300 4305 4310 4315 4320 4325 4330 4335 4340 4345 4350 4355 4360 4365 4370 4375 4380 4385 4390 4395 4400 4405 4410 4415 4420 4425 4430 4435 4440 4445 4450 4455 4460 4465 4470 4475 4480 4485 4490 4495 4500 4505 4510 4515 4520 4525 4530 4535 4540 4545 4550 4555 4560 4565 4570 4575 4580 4585 4590 4595 4600 4605 4610 4615 4620 4625 4630 4635 4640 4645 4650 4655 4660 4665 4670 4675 4680 4685 4690 4695 4700 4705 4710 4715 4720 4725 4730 4735 4740 4745 4750 4755 4760 4765 4770 4775 4780 4785 4790 4795 4800 4805 4810 4815 4820 4825 4830 4835 4840 4845 4850 4855 4860 4865 4870 4875 4880 4885 4890 4895 4900 4905 4910 4915 4920 4925 4930 4935 4940 4945 4950 4955 4960 4965 4970 4975 4980 4985 4990 4995 5000 5005 5010 5015 5020 5025 5030 5035 5040 5045 5050 5055 5060 5065 5070 5075 5080 5085 5090 5095 5100 5105 5110 5115 5120 5125 5130 5135 5140 5145 5150 5155 5160 5165 5170 5175 5180 5185 5190 5195 5200 5205 5210 5215 5220 5225 5230 5235 5240 5245 5250 5255 5260 5265 5270 5275 5280 5285 5290 5295 5300 5305 5310 5315 5320 5325 5330 5335 5340 5345 5350 5355 5360 5365 5370 5375 5380 5385 5390 5395 5400 5405 5410 5415 5420 5425 5430 5435 5440 5445 5450 5455 5460 5465 5470 5475 5480 5485 5490 5495 5500 5505 5510 5515 5520 5525 5530 5535 5540 5545 5550 5555 5560 5565 5570 5575 5580 5585 5590 5595 5600 5605 5610 5615 5620 5625 5630 5635 5640 5645 5650 5655 5660 5665 5670 5675 5680 5685 5690 5695 5700 5705 5710 5715 5720 5725 5730 5735 5740 5745 5750 5755 5760 5765 5770 5775 5780 5785 5790 5795 5800 5805 5810 5815 5820 5825 5830 5835 5840 5845 5850 5855 5860 5865 5870 5875 5880 5885 5890 5895 5900 5905 5910 5915 5920 5925 5930 5935 5940 5945 5950 5955 5960 5965 5970 5975 5980 5985 5990 5995 6000 6005 6010 6015 6020 6025 6030 6035 6040 6045 6050 6055 6060 6065 6070 6075 6080 6085 6090 6095 6100 6105 6110 6115 6120 6125 6130 6135 6140 6145 6150 6155 6160 6165 6170 6175 6180 6185 6190 6195 6200 6205 6210 6215 6220 6225 6230 6235 6240 6245 6250 6255 6260 6265 6270 6275 6280 6285 6290 6295 6300 6305 6310 6315 6320 6325 6330 6335 6340 6345 6350 6355 6360 6365 6370 6375 6380 6385 6390 6395 6400 6405 6410 6415 6420 6425 6430 6435 6440 6445 6450 6455 6460 6465 6470 6475 6480 6485 6490 6495 6500 6505 6510 6515 6520 6525 6530 6535 6540 6545 6550 6555 6560 6565 6570 6575 6580 6585 6590 6595 6600 6605 6610 6615 6620 6625 6630 6635 6640 6645 6650 6655 6660 6665 6670 6675 6680 6685 6690 6695 6700 6705 6710 6715 6720 6725 6730 6735 6740 6745 6750 6755 6760 6765 6770 6775 6780 6785 6790 6795 6800 6805 6810 6815 6820 6825 6830 6835 6840 6845 6850 6855 6860 6865 6870 6875 6880 6885 6890 6895 6900 6905 6910 6915 6920 6925 6930 6935 6940 6945 6950 6955 6960 6965 6970 6975 6980 6985 6990 6995 7000 7005 7010 7015 7020 7025 7030 7035 7040 7045 7050 7055 7060 7065 7070 7075 7080 7085 7090 7095 7100 7105 7110 7115 7120 7125 7130 7135 7140 7145 7150 7155 7160 7165 7170 7175 7180 7185 7190 7195 7200 7205 7210 7215 7220 7225 7230 7235 7240 7245 7250 7255 7260 7265 7270 7275 7280 7285 7290 7295 7300 7305 7310 7315 7320 7325 7330 7335 7340 7345 7350 7355 7360 7365 7370 7375 7380 7385 7390 7395 7400 7405 7410 7415 7420 7425 7430 7435 7440 7445 7450 7455 7460 7465 7470 7475 7480 7485 7490 7495 7500 7505 7510 7515 7520 7525 7530 7535 7540 7545 7550 7555 7560 7565 7570 7575 7580 7585 7590 7595 7600 7605 7610 7615 7620 7625 7630 7635 7640 7645 7650 7655 7660 7665 7670 7675 7680 7685 7690 7695 7700 7705 7710 7715 7720 7725 7730 7735 7740 7745 7750 7755 7760 7765 7770 7775 7780 7785 7790 7795 7800 7805 7810 7815 7820 7825 7830 7835 7840 7845 7850 7855 7860 7865 7870 7875 7880 7885 7890 7895 7900 7905 7910 7915 7920 7925 7930 7935 7940 7945 7950 7955 7960 7965 7970 7975 7980 7985 7990 7995 8000 8005 8010 8015 8020 8025 8030 8035 8040 8045 8050 8055 8060 8065 8070 8075 8080 8085 8090 8095 8100 8105 8110 8115 8120 8125 8130 8135 8140 8145 8150 8155 8160 8165 8170 8175 8180 8185 8190 8195 8200 8205 8210 8215 8220 8225 8230 8235 8240 8245 8250 8255 8260 8265 8270 8275 8280 8285 8290 8295 8300 8305 8310 8315 8320 8325 8330 8335 8340 8345 8350 8355 8360 8365 8370 8375 8380 8385 8390 8395 8400 8405 8410 8415 8420 8425 8430 8435 8440 8445 8450 8455 8460 8465 8470 8475 8480 8485 8490 8495 8500 8505 8510 8515 8520 8525 8530 8535 8540 8545 8550 8555 8560 8565 8570 8575 8580 8585 8590 8595 8600 8605 8610 8615 8620 8625 8630 8635 8640 8645 8650 8655 8660 8665 8670 8675 8680 8685 8690 8695 8700 8705 8710 8715 8720 8725 8730 8735 8740 8745 8750 8755 8760 8765 8770 8775 8780 8785 8790 8795 8800 8805 8810 8815 8820 8825 8830 8835 8840 8845 8850 8855 8860 8865 8870 8875 8880 8885 8890 8895 8900 8905 8910 8915 8920 8925 8930 8935 8940 8945 8950 8955 8960 8965 8970 8975 8980 8985 8990 8995 9000 9005 9010 9015 9020 9025 9030 9035 9040 9045 9050 9055 9060 9065 9070 9075 9080 9085 9090 9095 9100 9105 9110 9115 9120 9125 9130 9135 9140 9145 9150 9155 9160 9165 9170 9175 9180 9185 9190 9195 9200 9205 9210 9215 9220 9225 9230 9235 9240 9245 9250 9255 9260 9265 9270 9275 9280 9285 9290 9295 9300 9305 9310 9315 9320 9325 9330 9335 9340 9345 9350 9355 9360 9365 9370 9375 9380 9385 9390 9395 9400 9405 9410 9415 9420 9425 9430 9435 9440 9445 9450 9455 9460 9465 9470 9475 9480 9485 9490 9495 9500 9505 9510 9515 9520 9525 9530 9535 9540 9545 9550 9555 9560 9565 9570 9575 9580 9585 9590 9595 9600 9605 9610 9615 9620 9625 9630 9635 9640 9645 9650 9655 9660 9665 9670 9675 9680 9685 9690 9695 9700 9705 9710 9715 9720 9725 9730 9735 9740 9745 9750 9755 9760 9765 9770 9775 9780 9785 9790 9795 9800 9805 9810 9815 9820 9825 9830 9835 9840 9845 9850 9855 9860 9865 9870 9875 9880 9885 9890 9895 9900 9905 9910 9915 9920 9925 9930 9935 9940 9945 9950 9955 9960 9965 9970 9975 9980 9985 9990 9995 10000 10005 10010 10015 10020 10025 10030 10035 10040 10045 10050 10055 10060 10065 10070 10075 10080 10085 10090 10095 10100 10105 10110 10115 10120 10125 10130 10135 10140 10145 10150 10155 10160 10165 10170 10175 10180 10185 10190 10195 10200 10205 10210 10215 10220 10225 10230 10235 10240 10245 10250 10255 10260 10265 10270 10275 10280 10285 10290 10295 10300 10305 10310 10315 10320 10325 10330 10335 10340 10345 10350 10355 10360 10365 10370 10375 10380 10385 10390 10395 10400 10405 10410 10415 10420

including comparing the left and right context of a token of the token sequence with a set of predetermined token patterns to determine whether one of said set of predetermined left and right token patterns is satisfied; upon identifying the first token of the second subset of tokens, storing sequential tokens which follow the first token of the second subset of tokens until the last token of the second subset of tokens is identified.

9. A method of extracting a first subset of tokens having a first attribute and a second subset of tokens having a second attribute from a token sequence, the second subset of tokens appearing after the first subset of tokens, the first and second attribute selected from a group of predetermined attributes, the method comprising the steps of:

- identifying the first token of the first subset of tokens by applying a first set of rules to sequential ones of tokens in the token sequence;
- identifying the last token of second subset of tokens by applying a second set of rules to sequential ones of tokens after the first token of the first subset of tokens;
- identifying the last token of the first subset of tokens by applying a third set of rules to sequential ones of tokens after the first token of the first subset of tokens;
- identifying the first token of the second subset of tokens by applying a fourth set of rules to sequential ones of tokens after the last token of the first subset of tokens;
- identifying the attribute associated with the first subset of tokens; and
- identifying the attribute associated with the second subset of tokens.

10. The method of claim 9, wherein the first set of rules for identifying the first token of the first subset of tokens includes rules for comparing a left context and a right context of a token with a set of predetermined left and right token patterns to determine whether one of said set of predetermined left and right token patterns is satisfied.

11. A method of extracting attributes from a token sequence, each attribute comprising at least one token, the method comprising the steps of:

- accepting a plurality of rules which characterize a transition from a first attribute to a second attribute in the token sequence in terms of the left context and right context of a token;
- accepting the token sequence;
- identifying the borders between attributes in the token sequence by applying the plurality of rules to the token sequence.

12. The method of claim 11, wherein the transition from said first attribute to said second attribute includes a first transition from said first attribute to a dummy attribute, and a second transition from said dummy attribute to said second attribute.

13. The method of claim 11, wherein the attributes within the token sequence belong to a predetermined set of attributes, and said plurality of rules include rules for characterizing a transition from one attribute within the set of attributes to any other attribute within the set of attributes.

14. The method of claim 11, wherein the attributes within the token sequence belong to a predetermined set of attributes, and said plurality of rules include rules for characterizing a transition from one attribute within the set of attributes to a limited number of other attributes within the set of attributes.

15. A method of extracting subsets of tokens from a token sequence, each subset of tokens having an attribute selected from a group of attributes, the method comprising the steps of:

- accepting a plurality of rules that identifies the borders of different subsets of tokens having different attributes by matching the left context and right context of a token to a predetermined set of left and right token patterns;
- accepting the token sequence;
- identifying the start and end of subsets of tokens by applying the plurality of rules to sequential ones of the token sequence; and
- storing the subsets of tokens along with the attributes associated with the subset of tokens.

16. The method of claim 15, wherein the step of identifying the start and end of subsets of tokens comprises the steps of:

- applying a first subset of the plurality of rules to sequential ones of tokens in the token sequence to identify the start of the first subset of tokens and the end of the last subset of tokens;
- applying a second subset of the plurality of rules to sequential ones of the token sequence after the start of the first subset of tokens and before the end of the last subset of tokens, to identify the start and end of each subset of tokens;
- applying a third subset of the plurality of rules to identify the attributes associated with each subset of tokens.

17. A system for identifying attributes in a token se-

quence, comprising:

a storage device for storing a plurality of rules which characterize a transition from a first attribute to a second attribute in the token sequence in terms of the left context and right context of a token;

a processor for accepting the token sequence and identifying the first and last tokens of attributes in the token sequence by applying the plurality of rules to the token sequence.

18. The system of claim 15, wherein the plurality of rules characterize all possible transitions from a first attribute to a second attribute, and said processor applies the plurality of rules to sequential ones of the token sequence to identify the first and last token of each attribute in the token sequence within a single pass of reading the token sequence.

19. The system of claim 15, wherein the plurality of rules comprises three subsets of rule:

a first subset of rules for identifying the first token of the first attribute and last token of the last attribute in the token sequence;

a second subset of rules for identifying the first and last token of all attributes without actually identifying the individual attributes, and

a third subset of rules for identifying each individual attribute.

20. A method of generating a contextual rule used by an extractor for identifying transitions between subsets of tokens having different attributes, comprising the steps of:

receiving a sample token sequence including a first subset of tokens having a first attribute and a second subset of tokens not associated with the first attribute, the second subset of tokens coming before the first subset of tokens in the token sequence;

identifying the first token of the first subset of tokens;

generating a pair of left and right context patterns that match the left and right context of the first token of the first subset of tokens but not the left and right context of the tokens belonging to the second subset of tokens;

whereby the left and right context patterns form the matching criteria of the contextual rule for identifying the transition from the second subset of tokens to the first subset of tokens.

21. The method of claim 20, wherein the step of generating the pair of left and right context patterns comprises the steps of:

5

finding a first set of token classes that matches the first token of the right context of the first token of the first subset of tokens;

finding a second token classes that matches the first token of the left context of the first token of the first subset of tokens;

select a token class among the first set and second set of token classes such that the number of tokens in the second subset of tokens matched by said token class is the least;

assign said token class as part of the left context pattern if said token class matches the left context of the first token of the first subset of tokens; and

assign said token class as part of the right context pattern if said token class matches the right context of the first token of the first subset of tokens.

10

15

20 22. The method of claim 21, further comprising the steps of:

finding a third set of token classes that matches an additional token of the left context of the first token of the first subset of tokens that is not already matched by the left context pattern or the right context pattern;

finding a fourth of token classes that matches an additional token of the right context of the first token of the first subset of tokens that is not already matched by the left context pattern or the right context pattern;

select a token class among the third set and fourth set of token classes such that the number of tokens in the second subset of tokens matched by said token class is the least;

adding said token class as part of the left context pattern if said token class matches the left context of the first token of the first subset of tokens;

adding said token class as part of the right context pattern if said token class matches the right context of the first token of the first subset of tokens.

25

30

40

45

50

55

23. A method of generating a set of contextual rules used by an extractor for identifying the transitions between subsets of tokens having different attributes, the subsets of tokens being grouped in a plurality entries, comprising the steps of:

receiving a sample token sequence;

identifying each subset of tokens with corresponding attribute names;

grouping the subsets of tokens into separate entries; and

generating a contextual rule for each possible transitions between attributes;

wherein said contextual rule includes comparing the left and right context of a token to a pair of predetermined left and right context patterns, and a transition is identified when a match is found.

5

24. A system for generating contextual rules for identifying attributes in a token sequence, comprising:

a memory for storing a sample token sequence; 10
a user interface for allowing a user to identify at least one token within the sample token sequence that identifies the transition from a first attribute to a second attribute; 15
a processor for receiving the sample token sequence and generating a pair of left and right context patterns that match the left and right context of said at least one token that identifies the transition of attributes, whereby the left and right context patterns form the matching criteria 20
of a contextual rule for identifying the transition from said first attribute to said second attribute.

20

25. A method of generating transition rules for a single-pass extractor from a text sequence, the text sequence having a plurality of entries, each one of the plurality of entries having a set of attributes and a set of labels corresponding to the set of attributes, the method comprising the steps of:

25

(a) determining all possible permutations of attribute sequence within an entry from the set of attributes and set of labels in the text sequence; and 30
(b) creating a database of transition rules which identifies all possible transitions between attributes determined by step (a), each transition rule comprising the transitions from a first attribute to a dummy attribute and then to a second attribute.

35

40
45
26. A method of generating contextual rules for identifying transitions from a first subset of tokens to a second subset of tokens within a token sequence, the method comprising the steps of:

45

(a) identifying the positive token samples within the token sequence; 50
(b) identifying the negative token samples within the token sequence;
(c) applying a set-covering algorithm to generate a set of left and right context patterns that match the left and right context of all positive token samples and not match the left and right context of any negative token sample.

55

Faculty and Research Staff

- Mani Chandy
Professor of Computer Science and Executive Officer for Computer Science
- Taser Abu-Mostafa
Professor of Electrical Engineering and Computer Science
- Jim Arvo
Associate Professor of Computer Science
- John Tanner
Visiting Associate of Computer Science

Fig. 1 (a)

```
1:<H1>Faculty and Research Staff</H1>
2: <DT> <A HREF="http://www.cs.caltech.edu/people/mani.html"><B>Mani Chandy</B></A>
3:<DD> <I>Professor of Computer Science</I> and
4:   <I>Executive Officer for Computer Science</I>
5: <DT> <A HREF="http://electra.caltech.edu/EE/Faculty/Abu-Mostafa.html"><B>
       Taser Abu-Mostafa</B></A>
6:<DD> <I>Professor of Electrical Engineering and Computer Science</I>
7: <DT> <A HREF="http://www.cs.caltech.edu/~arvo/home.html"><B>Jim Arvo</B></A>
8:<DD> <I>Associate Professor of Computer Science</I>
9: <DT> <B>John Tanner</B>
10:<DD> <I>Visiting Associate of Computer Science</I>
```

Fig. 1 (b)

```
U = "http://www.cs.caltech.edu/people/mani.html"
N = "Mani Chandy"
A = "Professor of Computer Science"
M = "Executive Officer for Computer Science"

U = "http://electra.caltech.edu/EE/Faculty/Abu-Mostafa.html"
N = "Yaser Abu-Mostafa"
A = "Professor of Electrical Engineering and Computer Science"

U = "http://www.cs.caltech.edu/~arvo/home.html"
N = "Jim Arvo"
A = "Associate Professor of Computer Science"

N = "John Tanner"
A = "Visiting Associate of Computer Science"
```

Fig. 1 (c)

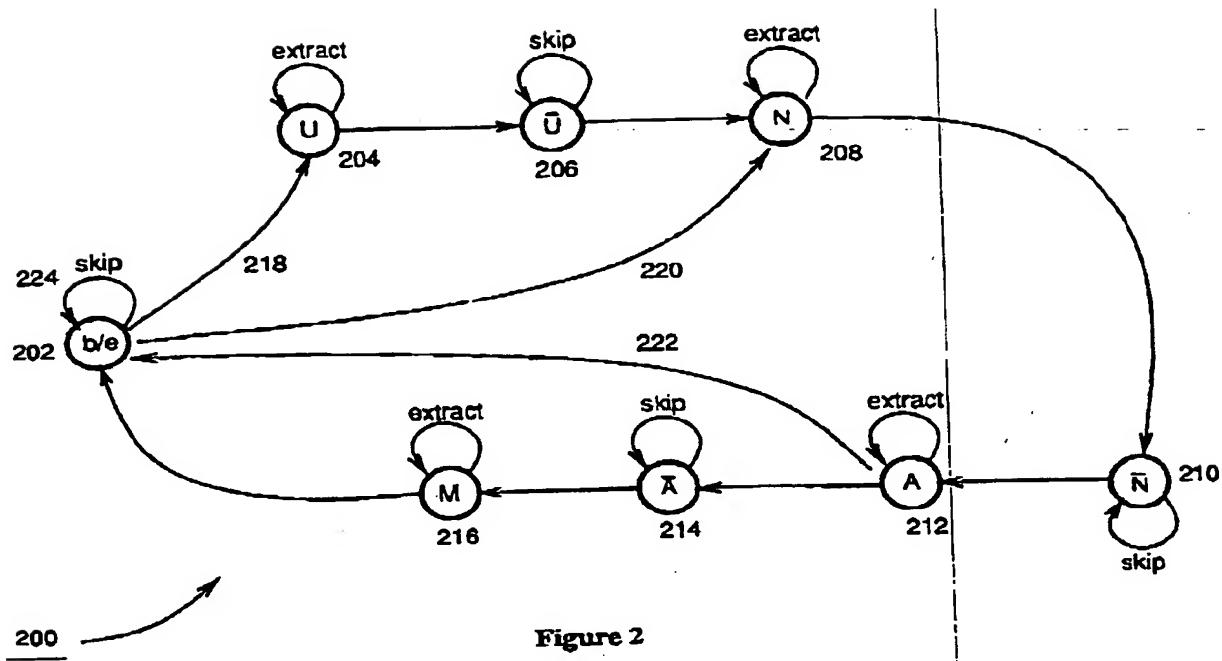


Figure 2

Fig. 2

$b/c \rightarrow U$
$b/c \rightarrow N$
$U \rightarrow U'$
$N \rightarrow N'$
$A \rightarrow A'$
$U' \rightarrow N$
$N' \rightarrow A$
$A' \rightarrow M$
$A \rightarrow b/e$
$M \rightarrow b/e$

Fig. 3

Token_Type	Pattern	Description	Length(byte)
CR	\r	The carriage return character.	Len=1
NL	\n	The new line character.	Len=1
TAB		The Tab character.	Len=1
PUNC	+ - / ?	Any symbol except A-Z, a-z, 0-9	Len=1
SPC	" "	The space character	Len=1
Calph	A ABC	All letters in a English word are capitalization	Len>=1
Clalph	Abc	All letters in a English word are lowercase letters except the fist one	Len>=2
Oalph	abc	All letters in a English word are lowercase letters	Len>=1
BIGS		A String begins with a Chinese word and ends with a Chinese word	Len>=2
Num	1.23 1,234,567 +1.23 +1,234,567 -1.23 -1,234,567 1*2+3 1/100	Single digit or any numeric expression contains digits or "+" "-" "/" "*" "," "." But single "+" "-" "/" "*" "," "." doesn't belongs to this type	Len>=1
Nalph	F16 16F 1F6	A String contains letters or digits But this string can't be a English word or a numeric expression	Len>=2
Html	<HTML> </HTML>	Any html tag	
Link	<a <img	Any hyperlink tag	

Fig. 4

1	2	3	4	5	6	7	8	9	10
<DT>		<A href=							
/	www	.	cs	.	caltech	.	edu	/	people
.									

11	12	13	14	15	16	17	18	19	20
/	Mani	.	html	"	>		Mani		Chandy
21	22	23	24	25	26	27	28	29	30

31	32	33							
		\n							

Fig. 5

```

<LI> <A href="http://www.cs.caltech.edu/people/mani.html">
  Mani Chandy</A>, <I>Professor of Computer Science</I> and
  <I>Executive Officer for Computer Science</I>
<LI> <A href="http://www.cs.caltech.edu/~arvo/home.html">
  Jim Arvo</A>, <I>Associate Professor of Computer Science</I>
<LI> <A href="http://www.ug.caltech.edu/~david/">
  David E. Breen</A>, <I>Assistant Director of Computer
  Graphics Laboratory</I>
<LI> John Tanner,
<I>Visiting Associate of Computer Science</I>

```

Fig. 6

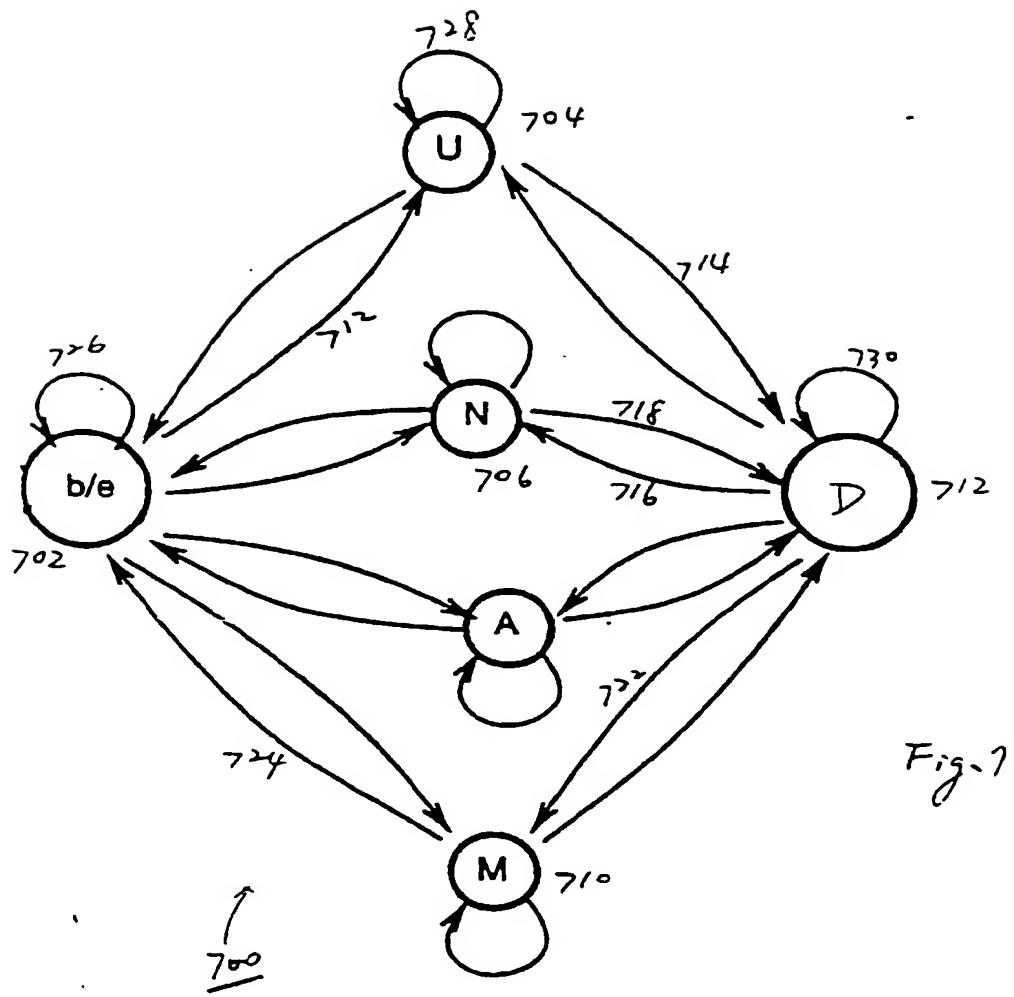


Fig. 7

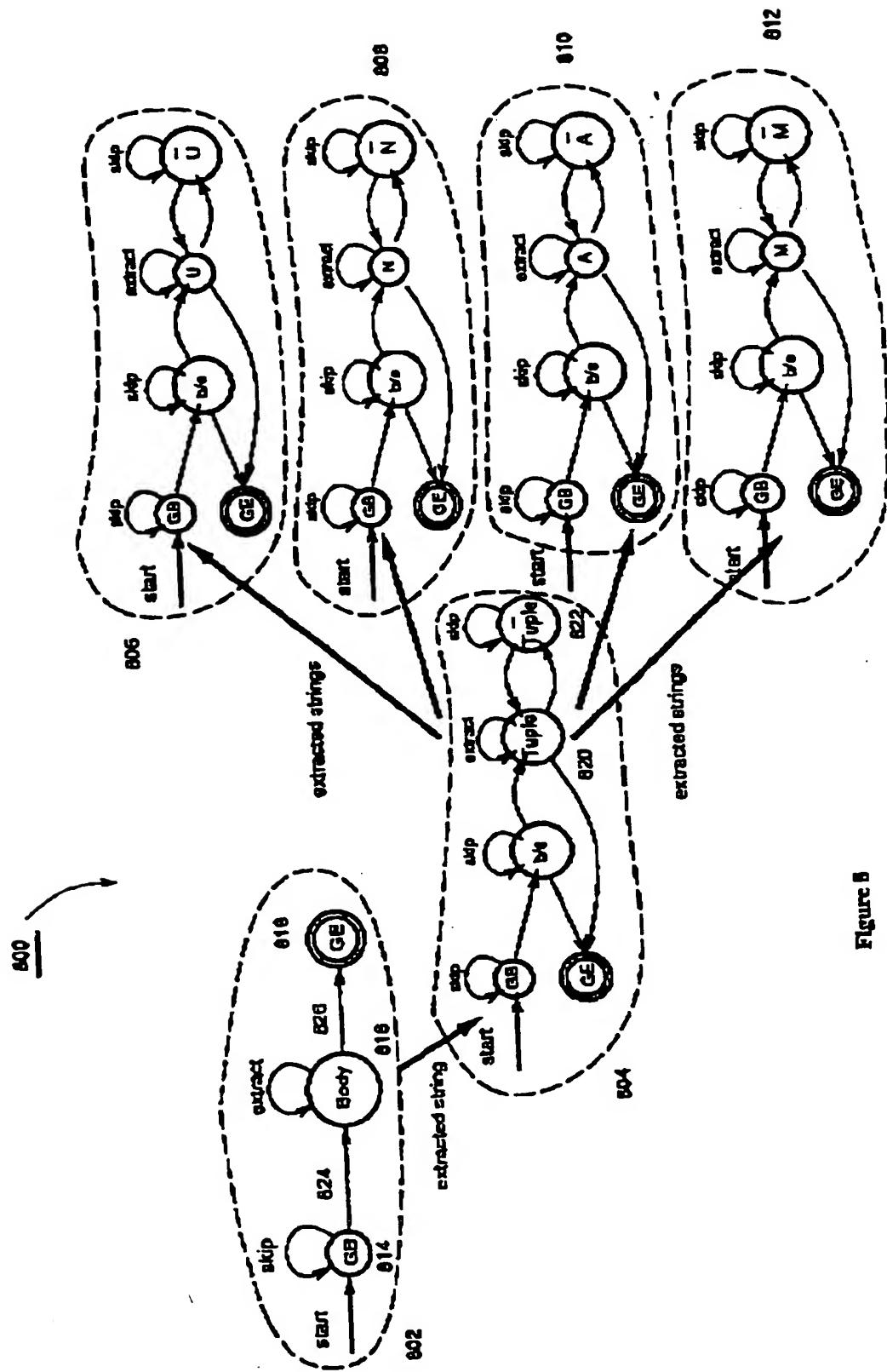


Figure 11

1:<H1>Faculty and Research Staff<H1>
2: <DT> Mani Chandy
3: <DD> <I>Professor of Computer Science</I> and
4: <I>Executive Officer for Computer Science</I>

5: <DT> Yaser Abu-Mostafa

6: <DD> <I>Professor of Electrical Engineering and Computer Science</I>

7: <DT> Jim Arvo

8: <DD> <I>Associate Professor of Computer Science</I>

9: <DT> John Tanner

10:<DD> <I>Visiting Associate of Computer Science</I>

Figure 9(a)

```
1:<H1>Faculty and Research Staff</H1>
2:<DT><A HREF="http://www.cs.caltech.edu/people/mani.html"><B>Mani Chandy</B></A>
3:<DD><I>Professor of Computer Science</I> and
4:<I>Executive Officer for Computer Science</I>
5:<DT><A HREF="http://electra.caltech.edu/EE/Faculty/Abu-Monstafa.html"><B>Yaser Abu-Mostafa</B></A>
6:<DD><I>Professor of Electrical Engineering and Computer Science</I>
7:<DT><A HREF="http://www.cs.caltech.edu/~arvo/home.html"><B>Jim Arvo</B></A>
8:<DD><I>Associate Professor of Computer Science</I>
9:<DT><B>John Tanner</B>
10:<DD><I>Visiting Associate of Computer Science</I>
```

Figure 9(b)

1.	Name:	'Lithium' J Smith
	E-Mail:	<u>animer@u.washington.edu</u>
	Last Update:	08/01/95
	Organization:	University of Washington
2.	Name:	'Sir Brand' Gregrobin
	Alt. Name:	Smith Gregrobin
	E-Mail:	<u>sirbrand@u.washington.edu</u>
	Organization:	university of washington
	Last Update:	06/21/96
	Organization:	University of Washington
3.	Name:	(raig Smith
	E-Mail:	<u>chs@maxwell.cs.uoregon.edu</u>
	Last Update:	08/01/94
	Organization:	University of Oregon
4.	Name:	- Richard Smith
	Alt. Name:	Richard
	E-Mail:	<u>GBORDERS@SFASUEDU</u>
	Last Update:	11/12/95
	Organization:	Stephen F. Austin State
		University
5.	Name:	- David S Smith
	Alt. Name:	David S
	E-Mail:	<u>dssmith@INDIANA.EDU</u>
	Last Update:	11/16/95
	Service Provider:	Indiana University

1000 →

Fig.10

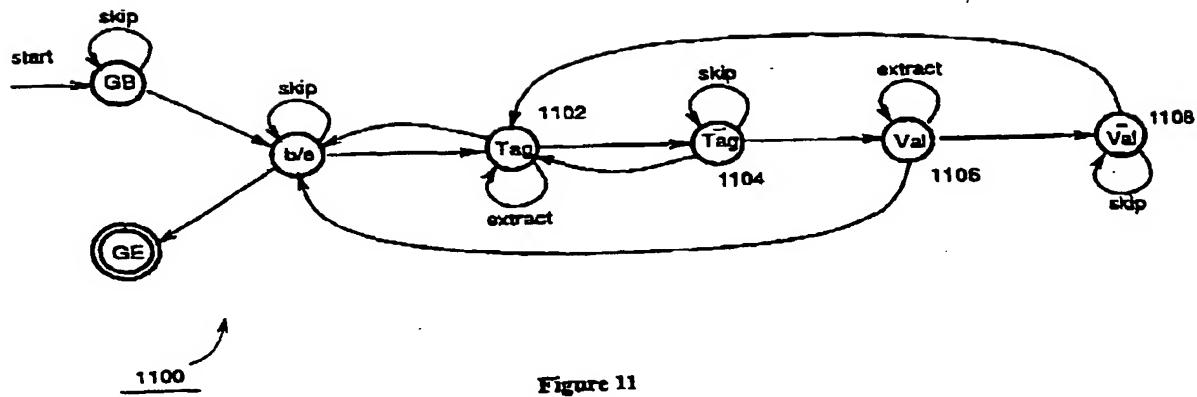


Figure 11

Name = "Lithium' J Smith"
E-Mail = aulmer@u.washington.edu
Last-Update = "08/01/95"
Organization = "University of Washington"

Fig. 12 (a)

Tag = "Name"
Value = "Lithium' J Smith"
Tag = "E-Mail"
Value = "aulmer@u.washington.edu"
Tag = "Last-Update"
Value = "08/01/95"
Tag = "Organization"
Value = "University of Washington"

Fig. 12 (b)

```

1: <H1>Faculty and Research Staff<H1>
2: <DT> <A HREF="http://www.cs.caltech.edu/people/mani.html"><B>Mani Chand</B></A>
3: <DD> <I>Professor of Computer Science</I> and
4: <I>Executive Officer for Computer Science</I>
5: <DT> <A HREF="http://electra.caltech.edu/EE/Faculty/Abu-Monstafa.html"><B>Yaser Abu-Mostafa</B></A>
6: <DD> <I>Professor of Electrical Engineering and Computer Science</I>
7: <DT> <A HREF="http://www.cs.caltech.edu/~arvo/home.html"><B>Jim Arvo</B></A>
8: <DD> <I>Associate Professor of Computer Science</I>
9: <DT> <B>John Tanner</B>
10:<DD> <I>Visiting Associate of Computer Science</I>

```

Fig. 13

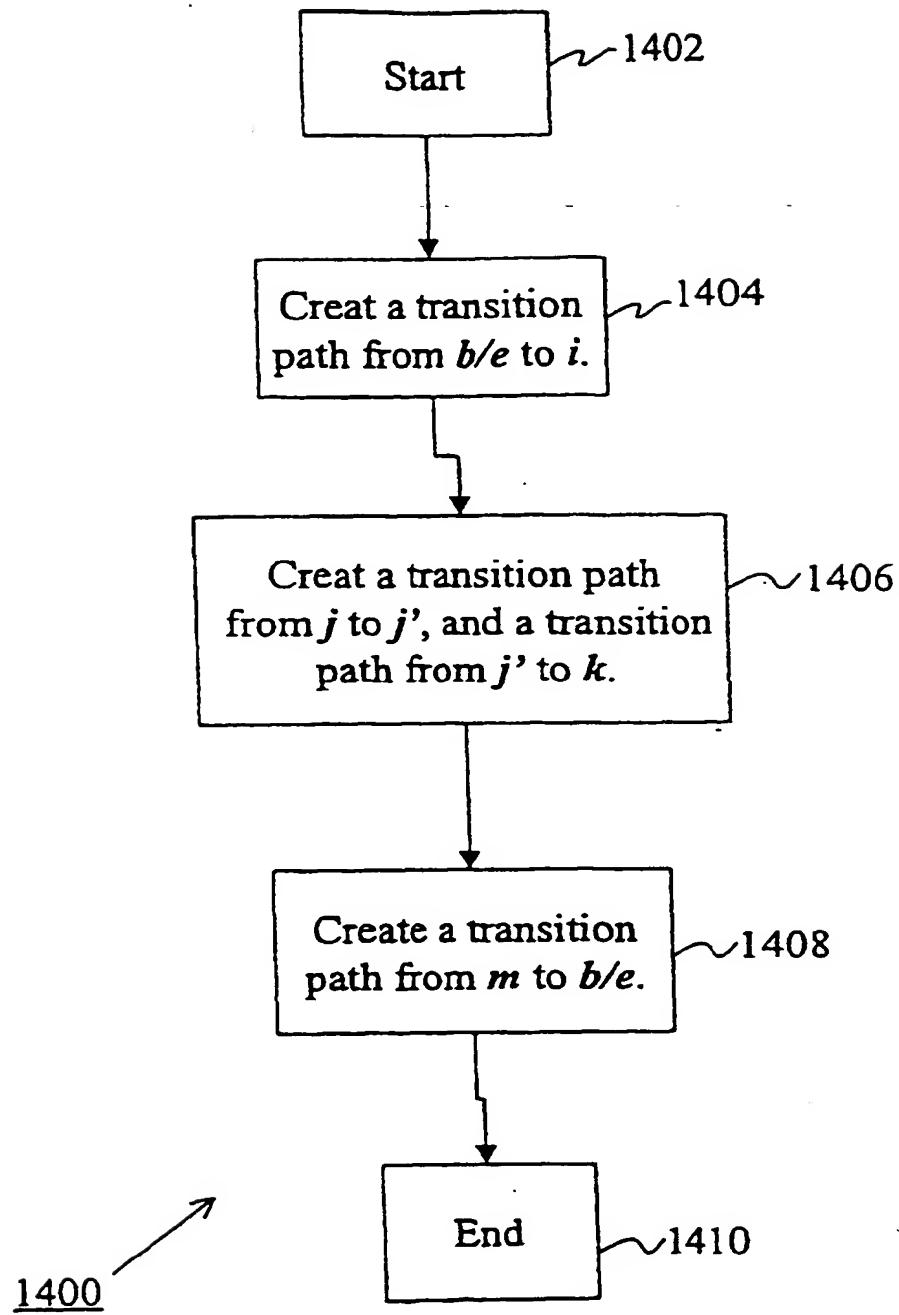


Fig. 14

A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
<H1>	Faculty		And		Research		Staff	</H1>	\n
A11	A12	A13	A14						A15
\r	<DT>								
A16	A17	A18		A19	A20				
Yaser		Abu-Mostafa							

Fig. 15 (a)

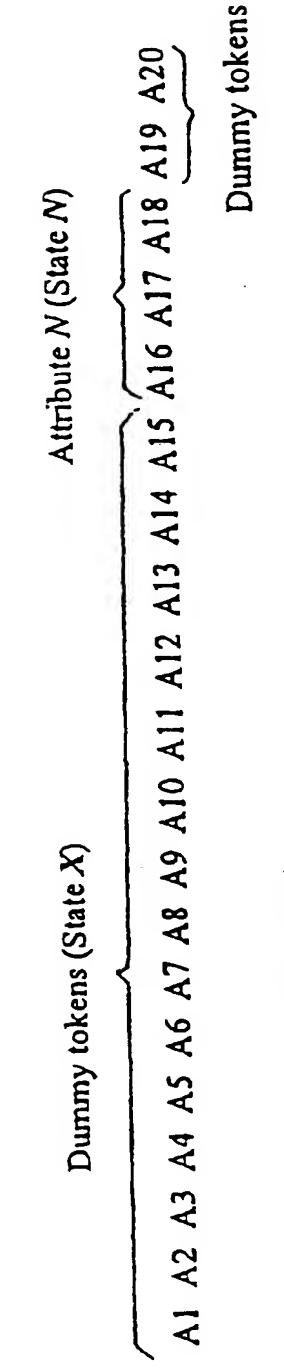


Fig. 15 (b)

	Left Context	Right Context
1 st token	A ₉	A ₁₀
2 nd token	A ₈	A ₁₁
3 rd token	A ₇	A ₁₂
i th token	A _(10-i)	A _{9+i}

Fig. 15 (c)

Token class	Definition
Word()	Includes token classes CNalph(), Big5(), Num(), and Ralph().
CNalph()	Includes token classes Calph(), Clalph(), and Dalph().
Calph()	Strings having only upper case letters; e.g., "ABC".
Clalph()	Strings having the first letter in uppercase, and all other letters in lower case; e.g., "Abc".
Oalph()	Strings having only lower case letter; e.g., "abc".
Big5()	Strings having characters in Big5 code (Chinese font code) and numbers.
Num()	Strings having numeric characters '0' to '9'.
Nalph()	Strings having at least one English alphabet and one numeric character; e.g., "F16".
Html()	Includes token classes Ptag() and Ttag().
Ptag()	HTML primitive tags; e.g., "<I>".
Ttag()	HTML tail tags; e.g., "</I>".
Nonword()	Includes token classes Punc() and Ctrl().
Punc()	A character that is not an alphabet or a number or a control character. E.g., any one of the following characters is a member of the token class Punc: ! ~ ! @ # \$ & ^ * () _ \ = - { } [] ; , . ? /).
Ctrl()	Includes token classes NC() and Spc().
NC()	Includes token classes Nl() and Cr().
Nl()	New line control character, "\n".
Cr()	Control character, "\r".
Spc()	Space character, " ".
Note: Here the term "string" refers to a sequence of alphanumeric characters without spaces in between.	

Fig. 16

	LP (left context pattern)	RP (right context pattern)	Matching count
Rule 6	Html()		P=1, n=4
Rule 7	Ptag()		P=1, n=4
Rule 8		Word()	P=1, n=4
Rule 9		CNalph()	P=1, n=4
Rule 10		C1alph()	P=1, n=3

Fig. 17 (a)

	LP (left context pattern)	RP (right context pattern)	Matching count
Rule 11	Html()	C1alph()	P=1, n=1
Rule 12	Ptag()	C1alph()	P=1, n=1
Rule 13		C1alph() Nonword()	P=1, n=2
Rule 14		C1alph() Ctrl()	P=1, n=2
Rule 15		C1alph() Spc()	P=1, n=2

Fig. 17 (b)

	LP (left context pattern)	RP (right context pattern)	Matching count
Rule 16	Html()Html()	C1alph()	P=1, n=0
Rule 17	Ptag()Html()	C1alph()	P=1, n=0
Rule 18	Html()	C1alph() Nonword()	P=1, n=1
Rule 19	Html()	C1alph() Ctrl()	P=1, n=1
Rule 20	Html()	C1alph() Spc()	P=1, n=1

Fig. 17 (c)

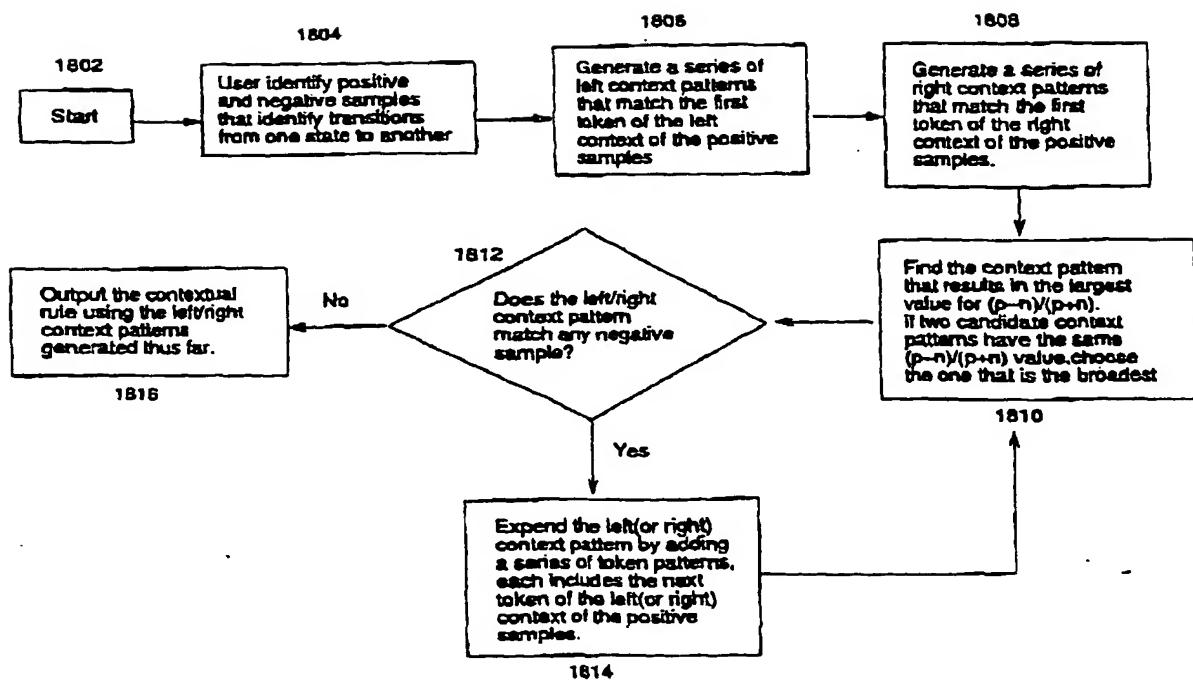


Figure 18

	LP (left context pattern)	RP (right context pattern)
Rule 21 (GB, b/e)	Html()Html()	Clalph()
Rule 22 (RB, Name)		Word()
Rule 23 (Name, @Name)		Html()
Rule 24 (@Name, Academic_title)		Clalph() Spc(1) 0alph()
Rule 25 (@Academic_title, Admin_title)		Clalph()
Rule 26 (Admin_title, b/e)		Html()
Rule 27 (Academic_title, b/e)		Ttag() NC()
Rule 28 (b/e, GE)	Spc(1) Clalph()	Spc(1) Nonword()

Fig. 19

```

1: <H1>Faculty and Research Staff<H1>
2: <DT> <A HREF="http://www.cs.caltech.edu/people/mani.html"><B>Mani Chandy</B></A>
   B1

3: <DD> <I>Professor of Computer Science</I> and
4: <I>Executive Officer for Computer Science</I>
5: <DT> <A HREF="http://electra.caltech.edu/EE/Faculty/Abu-Monstaafa.html"><B>Yaser Abu-Mostafa</B></A>
   B2

6: <DD> <I>Professor of Electrical Engineering and Computer Science</I>
7: <DT> <A HREF="http://www.cs.caltech.edu/~arvo/home.html"><B>Jim Arvo</B></A>
   B3

8: <DD> <I>Associate Professor of Computer Science</I>
9: <DT> <B>John Tanner</B>
   B4

10: <DD> <I>Visiting Associate of Computer Science</I>

```

Fig. 20

THIS PAGE BLANK (USPTO)